Clasificación

Taller de Procesamiento de Señales / Introducción a la Inteligencia Artificial

Agenda

Teoría de Clasificación

Bases

Objetivo: Clasificar Y (con $|\mathcal{Y}|$ finito) a partir del valor de X: $\hat{Y} = \varphi(X)$
Función costo: Hard $\rightarrow \quad \ell(x, y) = \mathbb{1}\{y \neq \varphi(x)\}$
Riesgo Esperado: Probabilidad de error $\rightarrow \quad \mathbb{P}(Y \neq \varphi(X))$

Teoría de Clasificación

Bases

Objetivo: Clasificar Y (con $|\mathcal{Y}|$ finito) a partir del valor de X: $\hat{Y} = \varphi(X)$
Función costo: Hard $\rightarrow \quad \ell(x, y) = \mathbb{1}\{y \neq \varphi(x)\}$
Riesgo Esperado: Probabilidad de error $\rightarrow \quad \mathbb{P}(Y \neq \varphi(X))$

Optimalidad

$$\mathbb{P}(Y \neq \varphi(X)) \geq 1 - \mathbb{E}\left[\max_y P_{Y|X}(y|X)\right]$$

con igualdad si y solo si $\varphi(x) = \arg\max_y P_{Y|X}(y|x)$.

Clasificador Bayesiano: $\varphi(x) = \arg\max_y P_{Y|X}(y|x)$

Error Bayesiano: $1 - \mathbb{E}\left[\max_y P_{Y|X}(y|X)\right]$

Clasificadores extremos

Clasificador bayesiano

El mejor clasificador (en términos de la probabilidad de error) es:

$$\mathbb{P}\left(Y \neq \varphi(X)\right) \geq 1 - \mathbb{E}\left[\max_{y} P_{Y|X}(y|X)\right]$$

Clasificador al azar para *k* clases

Cualquier clasificador razonable debe ganarle a la decisión al azar:

$$\mathbb{P}\left(Y \neq \varphi(X)\right) \leq 1 - \frac{1}{k}$$

Clasificador dummy

Otro clasificador muy precario (pero mejor que el azaroso) es elegir siempre la clase más probable. La probabilidad de error del dummy es:

$$\mathbb{P}\left(Y \neq \varphi(X)\right) \leq 1 - \max_{y} P_{Y}(y)$$

Clasificador bayesiano

Interpretación Gráfica

$$1 - \mathbb{E}\left[\max_y P_{Y|X}(y|X)\right] = \sum_{y \in \mathcal{Y}} P_Y(y)\mathbb{P}(X \notin \mathcal{R}_y | Y = y)$$

donde $\mathcal{R}_y$ es el conjunto de $x \in \mathcal{X}$ donde y es el máximo de $P_{Y|X=x}(y)$:

$$\mathcal{R}_y = \left\{x \in \mathcal{X} :\ P_{Y|X=x}(y) = \max_{y' \in \mathcal{Y}} P_{Y|X=x}(y')\right\}$$

Para los $x \in \mathcal{X}$ donde haya dos o más máximos de $P_{Y|X=x}(y)$, se asigna dicho x a solo una de dichas $\mathcal{R}_y$ elegida arbitrariamente.

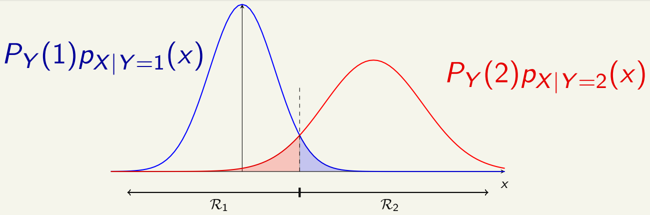
Clasificador bayesiano

Interpretación Gráfica

$$1 - \mathbb{E}\left[\max_y P_{Y|X}(y|X)\right] = \sum_{y \in \mathcal{Y}} P_Y(y)\mathbb{P}(X \notin \mathcal{R}_y | Y = y)$$

donde $\mathcal{R}_y$ es el conjunto de $x \in \mathcal{X}$ donde y es el máximo de $P_{Y|X=x}(y)$:

$$\mathcal{R}_y = \left\{x \in \mathcal{X} : \; P_{Y|X=x}(y) = \max_{y' \in \mathcal{Y}} P_{Y|X=x}(y')\right\}$$



Para los $x \in \mathcal{X}$ donde haya dos o más máximos de $P_{Y|X=x}(y)$, se asigna dicho x a solo una de dichas $\mathcal{R}_y$ elegida arbitrariamente.

Clasificador bayesiano

Objetivo

Quiero buscar $\varphi(\cdot)$ que minimice $\mathbb{P}(Y \neq \varphi(X))$. Es decir aprender el "clasificador bayesiano": $\varphi(x) = \arg\max_y P_{Y|X}(y|x)$.

Clasificador bayesiano

Objetivo

Quiero buscar $\varphi(\cdot)$ que minimice $\mathbb{P}(Y \neq \varphi(X))$. Es decir aprender el "clasificador bayesiano": $\varphi(x) = \arg\max_{y} P_{Y|X}(y|x)$.

Problemas numéricos

La propuesta de buscar $\varphi(\cdot)$ que minimice el riesgo empírico: $\frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{Y_i \neq \varphi(X_i)\}$ suele tener problemas numéricos (no derivable).

Clasificador bayesiano

Objetivo

Quiero buscar $\varphi(\cdot)$ que minimice $\mathbb{P}(Y \neq \varphi(X))$. Es decir aprender el "clasificador bayesiano": $\varphi(x) = \arg\max_y P_{Y|X}(y|x)$.

Problemas numéricos

La propuesta de buscar $\varphi(\cdot)$ que minimice el riesgo empírico: $\frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{Y_i \neq \varphi(X_i)\}$ suele tener problemas numéricos (no derivable).

Posible solución

El clasificador bayesiano se aprenderá en dos etapas:

- Aprender toda $P_{Y|X}(y|x)$.
- Quedarse con el máximo.

Divergencia de Kullback Leibler

Definición: Divergencia de Kullback Leibler

Sean $P(\cdot)$ y $Q(\cdot)$ dos funciones de masa de probabilidad tales que si $Q(y) = 0$ entonces $P(y) = 0$. Se define la divergencia KL como:

$$\mathrm{KL}(P\|Q) = \sum_{y\in\mathcal{Y}} P(y)\log\left(\frac{P(y)}{Q(y)}\right) = \mathbb{E}_P\left[\log\left(\frac{P(Y)}{Q(Y)}\right)\right]$$

donde el subíndice P en la esperanza hace referencia a que medida se utiliza para calcularla.

Divergencia de Kullback Leibler

Definición: Divergencia de Kullback Leibler

Sean $P(\cdot)$ y $Q(\cdot)$ dos funciones de masa de probabilidad tales que si $Q(y) = 0$ entonces $P(y) = 0$. Se define la divergencia KL como:

$$\mathrm{KL}(P\|Q) = \sum_{y\in\mathcal{Y}} P(y)\log\left(\frac{P(y)}{Q(y)}\right) = \mathbb{E}_P\left[\log\left(\frac{P(Y)}{Q(Y)}\right)\right]$$

donde el subíndice P en la esperanza hace referencia a que medida se utiliza para calcularla.

Teorema

$$\mathrm{KL}(P\|Q) \geq 0$$

con igualdad si y solo si $P(y) = Q(y)$ para todo $y \in \mathcal{Y}$.
(*Hint:* $\log(x) \leq x - 1$).

Divergencia de Kullback Leibler

Busco $\hat{P}(y|x)$ que minimice

$$\underbrace{\mathbb{E}\left[KL\left(P_{Y|X}(\cdot|X)\|\hat{P}(\cdot|X)\right)\right]}_{\text{Kullback Leibler}} = \underbrace{\mathbb{E}\left[-\log\hat{P}(Y|X)\right]}_{\text{Entropía Cruzada}} - \underbrace{\mathbb{E}\left[-\log P_{Y|X}(Y|X)\right]}_{\text{Entropía condicional}}$$

Divergencia de Kullback Leibler

Busco $\hat{P}(y|x)$ que minimice

$$\underbrace{\mathbb{E}\left[KL\left(P_{Y|X}(\cdot|X)\|\hat{P}(\cdot|X)\right)\right]}_{\text{Kullback Leibler}} = \underbrace{\mathbb{E}\left[-\log\hat{P}(Y|X)\right]}_{\text{Entropía Cruzada}} - \underbrace{\mathbb{E}\left[-\log P_{Y|X}(Y|X)\right]}_{\text{Entropía condicional}}$$

Optimalidad para $\ell(x,y) = -\log\hat{P}(y|x)$

$$\mathbb{E}\left[-\log\hat{P}(Y|X)\right] \geq \mathbb{E}\left[-\log P_{Y|X}(Y|X)\right]$$

son igualdad si y solo si $\hat{P}(y|x) = P_{Y|X}(y|x)$ para todo (x,y).

Divergencia de Kullback Leibler

Busco $\hat{P}(y|x)$ que minimice

$$\underbrace{\mathbb{E}\left[KL\left(P_{Y|X}(\cdot|X)\|\hat{P}(\cdot|X)\right)\right]}_{\text{Kullback Leibler}} = \underbrace{\mathbb{E}\left[-\log \hat{P}(Y|X)\right]}_{\text{Entropía Cruzada}} - \underbrace{\mathbb{E}\left[-\log P_{Y|X}(Y|X)\right]}_{\text{Entropía condicional}}$$

Optimalidad para $\ell(x,y) = -\log \hat{P}(y|x)$

$$\mathbb{E}\left[-\log \hat{P}(Y|X)\right] \geq \mathbb{E}\left[-\log P_{Y|X}(Y|X)\right]$$

son igualdad si y solo si $\hat{P}(y|x) = P_{Y|X}(y|x)$ para todo (x,y).

Entropías

- Entropía: $H(Y) = \mathbb{E}[-\log P_Y(Y)]$.
- Entropía Condicional: $H(Y|X) = \mathbb{E}[-\log P_{Y|X=x}(Y)]$.
- Entropía Cruzada: $\mathbb{E}\left[-\log \hat{P}(Y|X)\right]$.

Hard/Soft Decision

Mismatch de métricas

El mínimo de la cross entropy no tiene por que coincidir exactamente con el mínimo de la probabilidad de error. En general se mira la cross entropy para reducir el bias y la probabilidad de error para prevenir el overfitting.

Hard/Soft Decision

Mismatch de métricas

El mínimo de la cross entropy no tiene por que coincidir exactamente con el mínimo de la probabilidad de error. En general se mira la cross entropy para reducir el bias y la probabilidad de error para prevenir el overfitting.

Decisión Suave

Sea $x \in \mathbb{R}^d$, llamamos predicción *soft* de un algoritmo a la predicción de las probabilidades estimadas $\hat{P}(\cdot|x)$. Esta estimación es un vector de probabilidades de todas las clases posibles (no negativas y suman 1). Su desempeño se suele medir con la entropía cruzada $\mathbb{E}[-\log \hat{P}(Y|X)]$.

Decisión Dura

Sea $x \in \mathbb{R}^d$, llamamos predicción *hard* de un algoritmo a la predicción final de la clase estimada $\varphi(x)$ (estimación del valor de Y). Se la define como $\varphi(x) = \arg\max_{y \in \mathcal{Y}} \hat{P}(y|x)$ y su desempeño se suele medir con la probabilidad de acierto $\mathbb{P}(Y = \varphi(X))$.

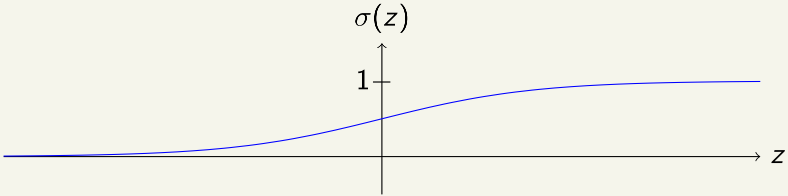
Outline

Regresión Logística Binaria

Función Sigmoide

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

- $\sigma(z)$ representa probabilidades.
- z recibe el nombre de *logit*.

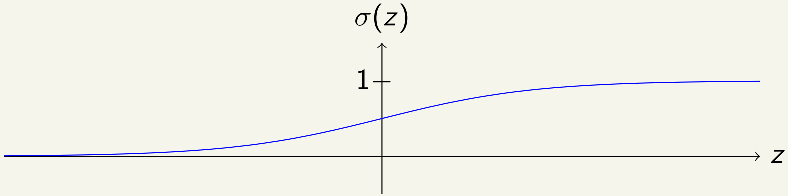


Regresión Logística Binaria

Función Sigmoide

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

- $\sigma(z)$ representa probabilidades.
- z recibe el nombre de *logit*.



Propuesta

$$\hat{P}(1|x) = \sigma(w^T x + b)$$
$$\hat{P}(0|x) = 1 - \sigma(w^T x + b)$$

Regresión Logística Binaria

Riesgo empírico

$$\frac{1}{n}\sum_{i=1}^{n}\ell(X_i, Y_i) =$$
$$-\frac{1}{n}\sum_{i=1}^{n} Y_i \log\left(\sigma(w^T X_i + b)\right) + (1 - Y_i)\log\left(1 - \sigma(w^T X_i + b)\right)$$

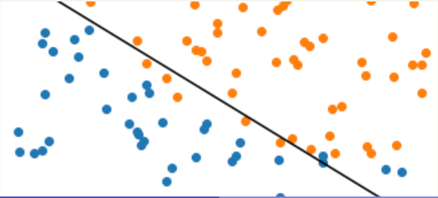
Regresión Logística Binaria

Riesgo empírico

$$\frac{1}{n}\sum_{i=1}^{n}\ell(X_i, Y_i) =$$
$$-\frac{1}{n}\sum_{i=1}^{n} Y_i \log\left(\sigma(w^T X_i + b)\right) + (1 - Y_i)\log\left(1 - \sigma(w^T X_i + b)\right)$$

Elección del máximo

$$\hat{P}(1|x) \lessgtr \hat{P}(0|x) \quad \Leftrightarrow \quad \hat{P}(1|x) \lessgtr \frac{1}{2} \quad \Leftrightarrow \quad w^T x + b \lessgtr 0$$



Métricas para clases desbalanceadas

Precision-Recall

Cuando el costo de las clases está desbalanceado se utilizan las métricas *Precision* y *Recall*.

- Precision $= \mathbb{P}(Y = \varphi(X)|\varphi(X) = 1)$. *Precision* se utiliza cuando los falsos positivos tiene consecuencias graves. Por ejemplo, diagnosticar erróneamente una enfermedad a una persona sana
- Recall (TPR) $= \mathbb{P}(Y = \varphi(X)|Y = 1)$. *Recall* se utiliza cuando cuando los falsos negativos tiene consecuencias graves. Por ejemplo, en la detección de fraudes, no detectar una transacción fraudulenta.

Métricas para clases desbalanceadas

Precision-Recall

Cuando el costo de las clases está desbalanceado se utilizan las métricas *Precision* y *Recall*.

- Precision $= \mathbb{P}(Y = \varphi(X) | \varphi(X) = 1)$. *Precision* se utiliza cuando los falsos positivos tiene consecuencias graves. Por ejemplo, diagnosticar erróneamente una enfermedad a una persona sana
- Recall (TPR) $= \mathbb{P}(Y = \varphi(X) | Y = 1)$. *Recall* se utiliza cuando cuando los falsos negativos tiene consecuencias graves. Por ejemplo, en la detección de fraudes, no detectar una transacción fraudulenta.

F1-score

Cuando la proporción de las clases está desbalanceada se utiliza la métrica F1:

$$F_1 = 2\frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Curvas ROC

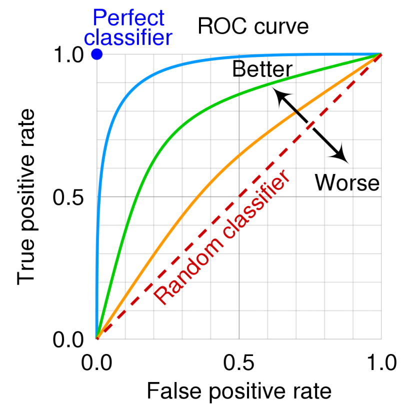
Agregar un umbral

Puedo darle más peso a una clase:

$$w^T x + b \lessgtr t$$

$$\text{TPR} = \mathbb{P}(Y = \varphi(X) | Y = 1)$$

$$\text{FPR} = \mathbb{P}(Y \neq \varphi(X) | Y = 0)$$



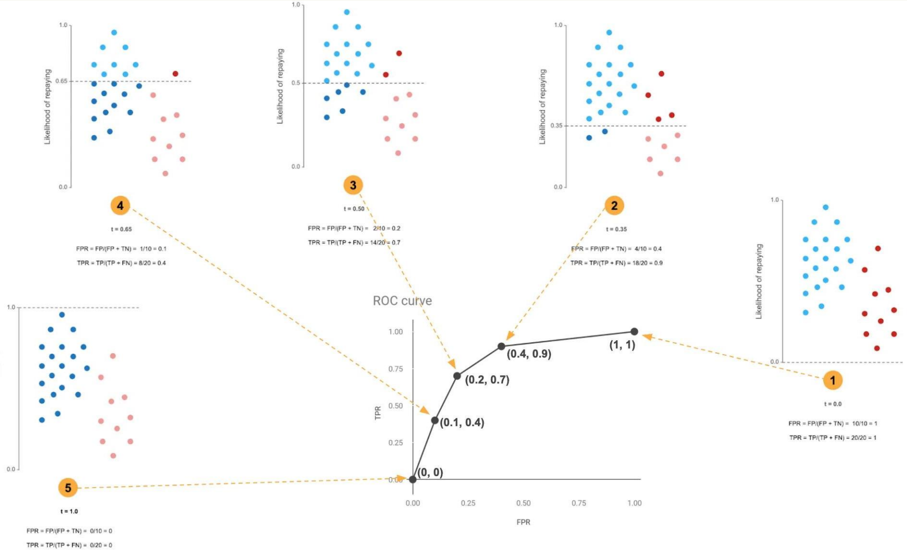
Area Under the Curve (AUC)

El AUC es el área bajo la curva ROC.

Equal Error Rate (EER)

El EER es el error para el cuál los errores $\text{FPR} = 1 - \text{TPR}$.

Curvas ROC



Likelihood of repaying
t = 0.65
FPR = FP/(FP + TN) = 1/10 = 0.1
TPR = TP/(TP + FN) = 8/20 = 0.4
t = 0.50
FPR = FP/(FP + TN) = 2/10 = 0.2
TPR = TP/(TP + FN) = 14/20 = 0.7
t = 0.35
FPR = FP/(FP + TN) = 4/10 = 0.4
TPR = TP/(TP + FN) = 18/20 = 0.9
t = 0.0
FPR = FP/(FP + TN) = 10/10 = 1
TPR = TP/(TP + FN) = 20/20 = 1
t = 1.0
FPR = FP/(FP + TN) = 0/10 = 0
TPR = TP/(TP + FN) = 0/20 = 0
ROC curve
TPR
FPR
(0, 0)
(0.1, 0.4)
(0.2, 0.7)
(0.4, 0.9)
(1, 1)

Outline

Regresión Logística Categórica (k clases)

Regresión logística clásica

$$\hat{P}(y|x) = \begin{cases} \frac{e^{w_y^T x + b_y}}{1+\sum_{j=1}^{k-1} e^{w_j^T x + b_j}} & y \in \{1, \cdots, k-1\} \\ \\ \frac{1}{1+\sum_{j=1}^{k-1} e^{w_j^T x + b_j}} & y = k \end{cases}$$

Regresión Logística Categórica (k clases)

Regresión logística clásica

$$\hat{P}(y|x) = \begin{cases} \dfrac{e^{w_y^T x + b_y}}{1+\sum_{j=1}^{k-1} e^{w_j^T x + b_j}} & y \in \{1, \cdots, k-1\} \\[2ex] \dfrac{1}{1+\sum_{j=1}^{k-1} e^{w_j^T x + b_j}} & y = k \end{cases}$$

Softmax

$$\hat{P}(y|x) = \frac{e^{w_y^T x + b_y}}{\sum_{j=1}^{k} e^{w_j^T x + b_j}}, \qquad y \in \{1, \cdots, k\}$$

Regresión Logística Categórica (k clases)

Regresión logística clásica

$$\hat{P}(y|x) = \begin{cases} \frac{e^{w_y^T x + b_y}}{1+\sum_{j=1}^{k-1} e^{w_j^T x + b_j}} & y \in \{1, \cdots, k-1\} \\ \frac{1}{1+\sum_{j=1}^{k-1} e^{w_j^T x + b_j}} & y = k \end{cases}$$

Softmax

$$\hat{P}(y|x) = \frac{e^{w_y^T x + b_y}}{\sum_{j=1}^{k} e^{w_j^T x + b_j}}, \qquad y \in \{1, \cdots, k\}$$

Riesgo empírico

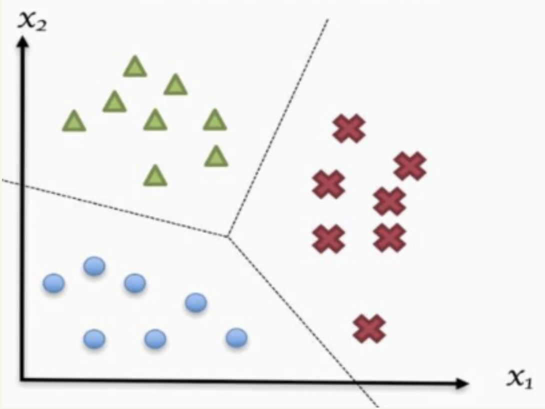
$$\frac{1}{n}\sum_{i=1}^{n} \ell(X_i, Y_i) = \frac{1}{n}\sum_{i=1}^{n} \left[\log\left(\sum_{j=1}^{k} e^{w_j^T X_i + b_j}\right) - \left(w_{Y_i}^T X_i + b_{Y_i}\right)\right]$$

Regresión Softmax

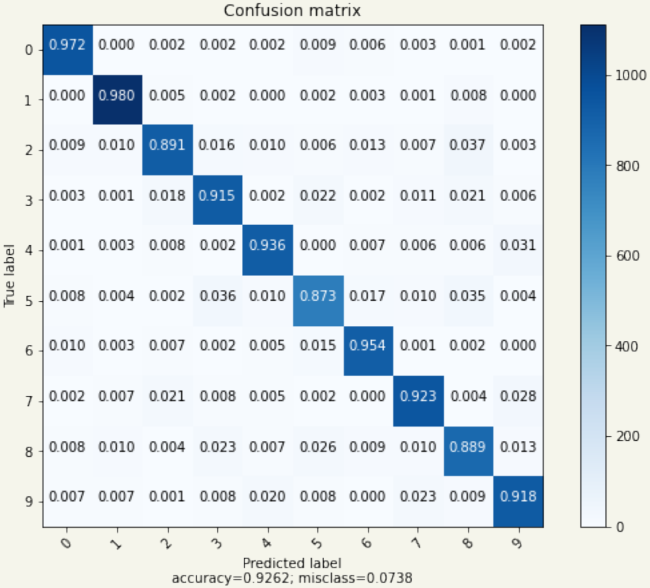
Elección del máximo

$$\arg\max_y \hat{P}(y|x) = \arg\max_y w_y^T x + b_y$$

Se separa con hiperplanos!



Confusion Matrix



Generalización del F1 score

		Predicted		
		Airplane	Boat	Car
Actual	Airplane	2	1	0
	Boat	0	1	0
	Car	1	2	3

Generalización del F1 score

		Predicted		
		Airplane	Boat	Car
Actual	Airplane	2	1	0
	Boat	0	1	0
	Car	1	2	3

Label	True Positive (TP)	False Positive (FP)	False Negative (FN)	Precision	Recall	F1 Score
Airplane	2	1	1	0.67	0.67	2 * (0.67 * 0.67) / (0.67 + 0.67) = **0.67**
Boat	1	3	0	0.25	1.00	2*(0.25 * 1.00) / (0.25 + 1.00) = **0.40**
Car	3	0	3	1.00	0.50	2 * (1.00 * 0.50) / (1.00 + 0.50) = **0.67**

Generalización del F1 score

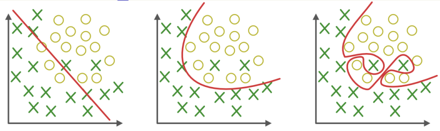
		Predicted		
		Airplane	Boat	Car
Actual	Airplane	2	1	0
	Boat	0	1	0
	Car	1	2	3

Label	True Positive (TP)	False Positive (FP)	False Negative (FN)	Precision	Recall	F1 Score
Airplane	2	1	1	0.67	0.67	2 * (0.67 * 0.67) / (0.67 + 0.67) = **0.67**
Boat	1	3	0	0.25	1.00	2*(0.25 * 1.00) / (0.25 + 1.00) = **0.40**
Car	3	0	3	1.00	0.50	2 * (1.00 * 0.50) / (1.00 + 0.50) = **0.67**

$$\text{Macro-}F_1 = \frac{0.67 + 0.40 + 0.67}{3} = 0.58$$

Outline

Overfitting/Underfitting



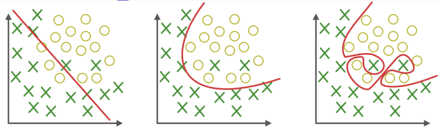
Regresión Logística Polinómica

El mapa polinómico se puede utilizar, combinandolo con su correspondiente regularización:

- $J(w,b) = \frac{1}{n_{\text{tr}}} \sum_{i=1}^{n} -\log \hat{P}(y_i|x_i) + \frac{\lambda}{n_{\text{tr}}} \|w\|^2$ (binario)
- $J(\theta) = \frac{1}{n_{\text{tr}}} \sum_{i=1}^{n} -\log \hat{P}(y_i|x_i) + \frac{\lambda}{n_{\text{tr}}} \sum_{j=1}^{K} \|w_j\|^2$ (categórico)

La etapa de validación se hace utilizando el *accuracy* como métrica.

Overfitting/Underfitting



Regresión Logística Polinómica

El mapa polinómico se puede utilizar, combinandolo con su correspondiente regularización:

- $J(w,b) = \frac{1}{n_{\text{tr}}} \sum_{i=1}^{n} -\log \hat{P}(y_i|x_i) + \frac{\lambda}{n_{\text{tr}}} \|w\|^2$ (binario)
- $J(\theta) = \frac{1}{n_{\text{tr}}} \sum_{i=1}^{n} -\log \hat{P}(y_i|x_i) + \frac{\lambda}{n_{\text{tr}}} \sum_{j=1}^{K} \|w_j\|^2$ (categórico)

La etapa de validación se hace utilizando el *accuracy* como métrica.

Calibración

La etapa de validación consiste en ajustar $\varphi(x)$ sin tener en cuenta las predicciones *soft*. Seguir considerando $\hat{P}(y|x)$ una estimación de la probabilidad puede ser problemático luego de una etapa exhaustiva de validación. Se denomina calibración a una etapa de re-ajuste de $\hat{P}(y|x)$.

Outline

Modelos Discriminativos y Generativos

Clasificación de Algoritmos

- **Modelos Discriminativos:** Modelan la dist. condicional $\hat{P}(y|x)$.
- **Modelos Generativos:** Modelan la dist. conjunta $\hat{P}(x, y)$.

Los modelos generativos permiten generar datos sintéticos!

Modelos Discriminativos y Generativos

Clasificación de Algoritmos

- **Modelos Discriminativos:** Modelan la dist. condicional $\hat{P}(y|x)$.
- **Modelos Generativos:** Modelan la dist. conjunta $\hat{P}(x, y)$.

Los modelos generativos permiten generar datos sintéticos!

Linear Discriminant Analysis (LDA)

$$Y \sim \text{Cat}(\{c_1, \cdots, c_K\}), \qquad X|Y = k \sim \mathcal{N}(\mu_k, \Sigma)$$

Linear Discriminant Analysis

Expresiones Matemáticas

$$\hat{p}(x) = \sum_{k=1}^{K} c_k \frac{e^{-\frac{1}{2}(x-\mu_k)^T \Sigma^{-1}(x-\mu_k)}}{(2\pi)^{d_x/2}|\Sigma|^{1/2}}$$

$$\hat{P}(y|x) = \frac{e^{\mu_y^T \Sigma^{-1} x - \frac{1}{2}\mu_y^T \Sigma^{-1}\mu_y + \log(c_y)}}{\sum_{k=1}^{K} e^{\mu_k^T \Sigma^{-1} x - \frac{1}{2}\mu_k^T \Sigma^{-1}\mu_k + \log(c_k)}}$$

Relación con Regresión Logística

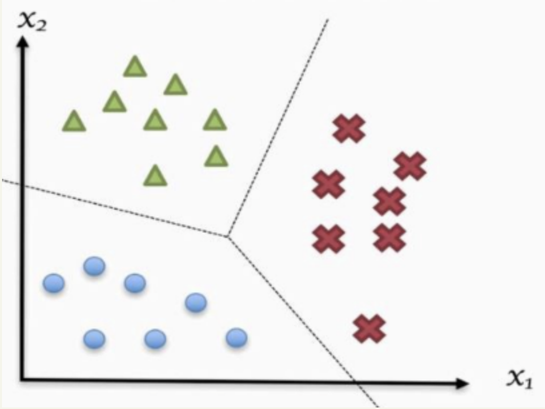
Si $w_y = \Sigma^{-1}\mu_y$ y $b_y = -\frac{1}{2}\mu_y^T \Sigma^{-1}\mu_y + \log(c_y)$, $\hat{P}(y|x)$ es el softmax. LDA utiliza hipótesis más fuertes ya que no solo asume $\hat{P}(y|x)$ softmax, sino también $\hat{p}(x)$ mezcla de gaussianas.

Regresión Softmax

Elección del máximo

$$\arg\max_y \hat{P}(y|x) = \arg\max_y w_y^T x + b_y$$

Se separa con hiperplanos!



Estimación Insesgada de Parámetros

Estimadores

$$\mathcal{D}_k = \{x_i : \ 1 \leq i \leq n \ \wedge \ y_i = k\}$$

$$c_k = \frac{\#(\mathcal{D}_k)}{n}$$

$$\mu_k = \frac{1}{\#(\mathcal{D}_k)} \sum_{x \in \mathcal{D}_k} x$$

$$\Sigma_k = \frac{1}{\#(\mathcal{D}_k) - 1} \sum_{x \in \mathcal{D}_k} (x - \mu_k)(x - \mu_k)^T$$

$$\Sigma = \frac{1}{n - K} \sum_{k=1}^{K} (\#(\mathcal{D}_k) - 1) \Sigma_k$$

Quadratic Discriminant Analysis

Quadratic Discriminant Analysis (QDA)

$$Y \sim \mathrm{Cat}(\{c_1, \cdots, c_K\}), \qquad X|Y = k \sim \mathcal{N}(\mu_k, \Sigma_k)$$

Quadratic Discriminant Analysis

Quadratic Discriminant Analysis (QDA)

$$Y \sim \mathrm{Cat}(\{c_1, \cdots, c_K\}), \qquad X|Y = k \sim \mathcal{N}(\mu_k, \Sigma_k)$$

Expresiones Matemáticas

$$\hat{p}(x) = \sum_{k=1}^{K} c_k \frac{e^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)}}{(2\pi)^{d_x/2}|\Sigma_k|^{1/2}}$$

$$\hat{P}(y|x) = \frac{e^{-\frac{1}{2}(x-\mu_y)^T \Sigma_y^{-1}(x-\mu_y)+\log(c_y)-\frac{\log|\Sigma_y|}{2}}}{\sum_{k=1}^{K} e^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)+\log(c_k)-\frac{\log|\Sigma_k|}{2}}}$$

Quadratic Discriminant Analysis

Quadratic Discriminant Analysis (QDA)

$$Y \sim \mathrm{Cat}(\{c_1, \cdots, c_K\}), \qquad X|Y = k \sim \mathcal{N}(\mu_k, \Sigma_k)$$

Expresiones Matemáticas

$$\hat{p}(x) = \sum_{k=1}^{K} c_k \frac{e^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)}}{(2\pi)^{d_x/2}|\Sigma_k|^{1/2}}$$

$$\hat{P}(y|x) = \frac{e^{-\frac{1}{2}(x-\mu_y)^T \Sigma_y^{-1}(x-\mu_y)+\log(c_y)-\frac{\log|\Sigma_y|}{2}}}{\sum_{k=1}^{K} e^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)+\log(c_k)-\frac{\log|\Sigma_k|}{2}}}$$

Elección del máximo: NO ES LINEAL, ES CUADRÁTICO

$$\arg\max_y \; -\frac{1}{2}(x-\mu_y)^T \Sigma_y^{-1}(x-\mu_y) + \log(c_y) - \frac{\log|\Sigma_y|}{2}$$

Resumen

Verosimilitud

- LDA: $X|Y = k \sim \mathcal{N}(\mu_k, \Sigma)$.
- QDA: $X|Y = k \sim \mathcal{N}(\mu_k, \Sigma_k)$.

Función discriminante

- LDA: $\delta_y(x) = \mu_y^T \Sigma^{-1} x - \frac{1}{2}\mu_y^T \Sigma^{-1} \mu_y + \log(c_y)$.
- QDA: $\delta_y(x) = -\frac{1}{2}(x - \mu_y)^T \Sigma_y^{-1}(x - \mu_y) + \log(c_y) - \frac{\log|\Sigma_y|}{2}$.

Prediction

- Decisión dura: $\varphi(x) = \arg\max_{y \in \mathcal{Y}} \delta_y(x)$.
- Decisión suave: $\hat{P}(y|x) = \operatorname{soft}\max_{y \in \mathcal{Y}} \delta_y(x)$.

Outline

Modelos Paramétricos y No Paramétricos

Clasificación de Algoritmos

- **Modelos Paramétrcios:** Asumen conocimiento parcial sobre la distribución, indexándola por parámetros.
- **Modelos No Paramétricos:** No se asume una estructura a priori para la distribución.

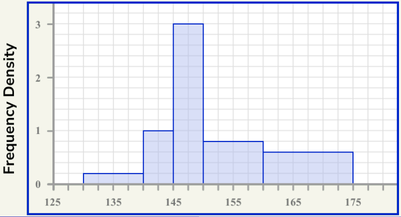
Modelos Paramétricos y No Paramétricos

Clasificación de Algoritmos

- **Modelos Paramétrcios:** Asumen conocimiento parcial sobre la distribución, indexándola por parámetros.
- **Modelos No Paramétricos:** No se asume una estructura a priori para la distribución.

Histograma

El histograma asume una densidad constante por regiones. En cada región asigna $\hat{p}(x) = \frac{K}{n \cdot V}$ donde n es la cantidad de muestras totales, K la cantidad de muestras en dicha región y V el volumen de la región.



K-Vecinos más cercanos (KNN)

Adaptando el concepto a aprendizaje supervisado

Asumiendo que $\hat{P}(y) = \frac{N_y}{n}$ con N_y el número de muestras de la clase y, y que (en cada región) $\hat{p}(x|y) = \frac{K_y}{N_y \cdot V}$ con K_y la cantidad de muestras que caen en la región de la clase y, se obtiene:

$$\hat{P}(y|x) = \frac{\hat{p}(x|y)\hat{P}(y)}{\sum_{i \in \mathcal{Y}} \hat{p}(x|i)\hat{P}(i)} = \frac{K_y}{K}$$

donde K es la cantidad de muestras totales en la región. Es decir, la proporción de muestras de la clase y en la región.

K-Vecinos más cercanos (KNN)

Adaptando el concepto a aprendizaje supervisado

Asumiendo que $\hat{P}(y) = \frac{N_y}{n}$ con N_y el número de muestras de la clase y, y que (en cada región) $\hat{p}(x|y) = \frac{K_y}{N_y \cdot V}$ con K_y la cantidad de muestras que caen en la región de la clase y, se obtiene:

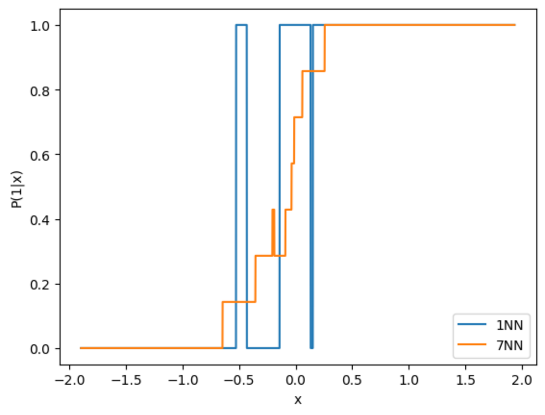
$$\hat{P}(y|x) = \frac{\hat{p}(x|y)\hat{P}(y)}{\sum_{i \in \mathcal{Y}} \hat{p}(x|i)\hat{P}(i)} = \frac{K_y}{K}$$

donde K es la cantidad de muestras totales en la región. Es decir, la proporción de muestras de la clase y en la región.

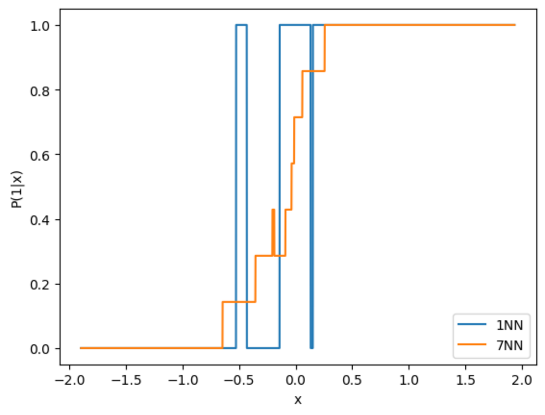
K-Vecinos más cercanos

KNN fija el valor de vecinos K y en base a esto define las regiones. Por ejemplo, la región utilizada para computar un *feature* x es la región centrada en x que posee K muestras (las K más cercanas a x).

K-Vecinos más cercanos (KNN)



K-Vecinos más cercanos (KNN)



Elección del máximo

Notar que para quedarse con el máximo de $\hat{P}(y|x)$ no hace falta computarla. Simplemente se clasifica según sus K vecinos más cercanos, por mayoría.

Outline

Support Vector Machines

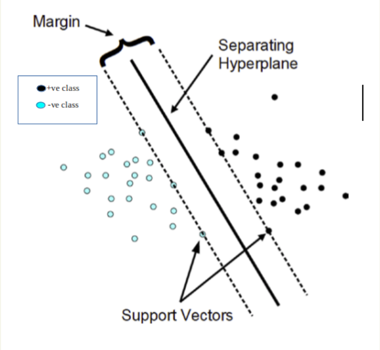
Clases linealmente separables

Sea la clasificación binaria $y \in \{-1, 1\}$ y $z(x) = w^T \cdot x + b = 0$ su frontera de decisión. Decimos que las clases son linealmente separables, si existen w y b tales que $y \cdot z(x) > 0$ para todo $(x, y) \in \mathcal{D}_n$ (set de entrenamiento). Llamamos $f_i(w, b) = y_i z(x_i) > 0$ con $1 \leq i \leq n$.

Support Vector Machines

Clases linealmente separables

Sea la clasificación binaria $y \in \{-1, 1\}$ y $z(x) = w^T \cdot x + b = 0$ su frontera de decisión. Decimos que las clases son linealmente separables, si existen w y b tales que $y \cdot z(x) > 0$ para todo $(x, y) \in \mathcal{D}_n$ (set de entrenamiento). Llamamos $f_i(w, b) = y_i z(x_i) > 0$ con $1 \leq i \leq n$.



- w es ortogonal a la frontera y por lo tanto $w//(x - x_*)$ con x_* la proyección ortogonal de x sobre la frontera.

$$|w^T(x - x_*)| = \|w\|\|x - x_*\|$$

- Dado que x_* está sobre la frontera, $w^T(x - x_*) = z(x)$ y por lo tanto:

$$d(x_i) = \|x_i - x_*\| = \frac{|z(x_i)|}{\|w\|} = \frac{y_i \cdot z(x_i)}{\|w\|}$$

Support Vector Machines

Margen

Se define el margen unilateral como criterio de peor caso:

$$m(w,b) = \min_{1\leq i\leq n} \frac{y_i(w^T \cdot x_i + b)}{\|w\|} = \frac{1}{\|w\|} \min_{1\leq i\leq n} f_i(w,b) = \frac{f_k(w,b)}{\|w\|}$$

con k un índice óptimo (función de w y b). Por lo tanto, el problema a resolver es maximizar el margen: $\max_{w,b}\ m(w,b)$ st. $f_i(w,b) > 0$ para $i = 1, \cdots, n$.

Support Vector Machines

Margen

Se define el margen unilateral como criterio de peor caso:

$$m(w, b) = \min_{1 \leq i \leq n} \frac{y_i(w^T \cdot x_i + b)}{\|w\|} = \frac{1}{\|w\|} \min_{1 \leq i \leq n} f_i(w, b) = \frac{f_k(w, b)}{\|w\|}$$

con k un índice óptimo (función de w y b). Por lo tanto, el problema a resolver es maximizar el margen: $\max_{w,b} \; m(w, b)$ st. $f_i(w, b) > 0$ para $i = 1, \cdots, n$.

Escala

Sea $\alpha > 0$, está claro la decisión $z(x) \gtrless 0$ no se ve afectada si reescalamos los parámetros $w \leftarrow \alpha w$ y $b \leftarrow \alpha b$. Esto mismo ocurre con el margen $m(\alpha w, \alpha b) = m(w, b)$. Con lo cuál no se pierde generalidad al asumir $f_k(w, b) = 1$. Luego $m(w, b) = \frac{1}{\|w\|}$ y $f_i(w, b) \geq 1$ para todo $1 \leq i \leq n$.

Las muestras en las que $f_i(w, b) = 1$ se denominan vectores soporte.

Support Vector Machines

Problema de optimización primal

$$\min_{w,b} \frac{1}{2}\|w\|^2 \quad \text{s.t.} \quad y_i(w^T x_i + b) \geq 1 \quad (\forall\, 1 \leq i \leq n)$$

Bishop - "Pattern Recognition and Machine Learning" Capítulo 7.

Support Vector Machines

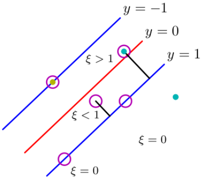
Problema de optimización primal

$$\min_{w,b} \frac{1}{2}\|w\|^2 \quad \text{s.t.} \quad y_i(w^T x_i + b) \geq 1 \quad (\forall\, 1 \leq i \leq n)$$

Relajando los márgenes

Mitigar problemas con outliers. Sea $C \geq 0$,

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \quad \text{s.t.} \quad \begin{cases} y_i(w^T x_i + b) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{cases} \quad (\forall\, 1 \leq i \leq n)$$

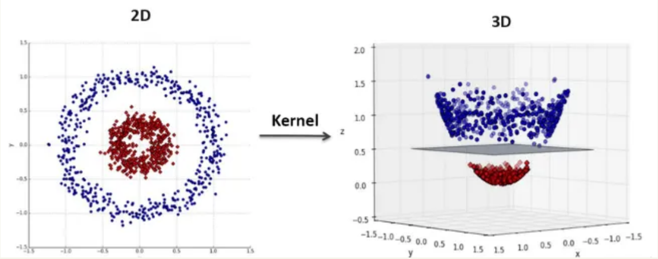


Bishop - "Pattern Recognition and Machine Learning" Capítulo 7.

Support Vector Machines

Generalización a fronteras no lineales

Este método es adaptable a diferentes fronteras $z(x) = w^T \phi(x) + b$. Se puede demostrar, que el resultado final del entrenamiento depende de los predictores a través de $k(x_1, x_2) = \phi^T(x_1)\phi(x_2)$, función que recibe el nombre de kernel. Es por este motivo que se elige el kernel en lugar de $\phi(\cdot)$, siendo el más utilizado en SVM el denominado gaussiano o rbf: $k(x_1, x_2) = e^{-\gamma \|x_1 - x_2\|^2}$.



Support Vector Machines

Generalización a K-clases

- *one-vs-one*: Se toman todas las combinaciones de pares de clases (son $\frac{K(K-1)}{2}$) y se entrenan clasificadores binarios. Se clasifica seleccionando a la clase con más *votos*.
- *one-vs-the-rest*: Se entrenan K clasificadores binarios, donde cada uno toma una clase como positiva y el resto como negativa. Se clasifica según $\arg\max_k w_k^T \phi(x) + b_k$.

Support Vector Machines

Generalización a K-clases

- *one-vs-one*: Se toman todas las combinaciones de pares de clases (son $\frac{K(K-1)}{2}$) y se entrenan clasificadores binarios. Se clasifica seleccionando a la clase con más *votos*.
- *one-vs-the-rest*: Se entrenan K clasificadores binarios, donde cada uno toma una clase como positiva y el resto como negativa. Se clasifica según $\arg\max_k w_k^T \phi(x) + b_k$.

Generalización a Regresión

$$\min_{w,b} \frac{1}{2}\|w\|^2 \quad \text{s.t.} \quad |w^T x_i + b - y_i| \leq \epsilon \quad (\forall\, 1 \leq i \leq n)$$

Optimización Convexa

Tomemos el problema básico de SVM. Sea

$$J_1 = \min_{w,b} \frac{1}{2}\|w\|^2 \quad \text{s.t.} \quad y_i(w^T\phi(x_i)+b) \geq 1 \quad (\forall\, 1 \leq i \leq n)$$

Dicho problema puede reescribirse usando multiplicadores de Lagrange α_i:

$$J_1 = \min_{w,b} \max_{\alpha_i \geq 0} \ \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i [y_i(w^T\phi(x_i)+b) - 1]$$

Vectores Soportes

Notar que, el multiplicador óptimo (la solución del problema) debe cumplir que $\alpha_i = 0$ para toda muestra que no sea vector soporte. En contraste, para los vectores soporte ocurre que $y_i(w^T\phi(x_i)+b) = 1$.

Llamamos problema dual al problema definido a partir de invertir el mínimo y el máximo:

$$J_2 = \max_{\alpha_i \geq 0} \min_{w,b} \ \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i [y_i(w^T\phi(x_i)+b) - 1]$$

Optimización Convexa

Teorema: Weak and Strongh duality

Para cualquier problema de optimización $J_1 \geq J_2$. En el caso particular del problema de SVM, por ser convexo, se obtiene que $J_1 = J_2$.

Fijo los multiplicadores $\alpha_i \geq 0$, igualamos a cero la derivada respecto de los parámetros para buscar el mínimo:

- $w - \sum_{i=1}^{n} \alpha_i y_i \phi(x_i) = 0 \quad \rightarrow \quad w = \sum_{i=1}^{n} \alpha_i y_i \phi(x_i)$
- $-\sum_{i=1}^{n} \alpha_i y_i = 0 \quad \rightarrow \quad \sum_{i=1}^{n} \alpha_i y_i = 0$

La suma dentro de J_2 queda como:

$$\sum_{i=1}^{n} \alpha_i [y_i (w^T \phi(x_i) + b) - 1] = \sum_{i=1}^{n} \alpha_i y_i w^T \phi(x_i) + \sum_{i=1}^{n} \alpha_i y_i b - \sum_{i=1}^{n} \alpha_i$$

$$= \|w\|^2 + 0 - \sum_{i=1}^{n} \alpha_i$$

Optimización Convexa

La función a optimizar se puede reescribir como

$$\frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i [y_i(w^T \phi(x_i) + b) - 1] = \left(\sum_{i=1}^{n} \alpha_i \right) - \frac{1}{2}\|w\|^2$$

La norma cuadrática puede vectorizarse como

$$\|w\|^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \phi(x_i)^T \phi(x_j) = \alpha^T Q \alpha$$

donde Q es una matriz de elementos $Q_{i,j} = y_i y_j \phi(x_i)^T \phi(x_j)$ (depende de los predictores a través del kernel). Entonces, el problema se reduce a:

Problema de optimización dual

$$\max_{\alpha} \ \alpha^T \mathbf{1} - \frac{1}{2}\alpha^T Q \alpha \quad \text{s.t.} \quad \alpha^T y = 0, \ \alpha_i \geq 0$$

Por cuestiones numéricas, suele traer complicaciones detectar vectores soportes como $\alpha_i > 0$. En la práctica suele compararse $\alpha_i > \epsilon$ con $\epsilon > 0$ un número pequeño.

Optimización Convexa

Bias

Sea $\mathcal{S}$ el conjunto de índices de vectores soporte y N_S la cantidad de elementos de dicho conjunto; luego $y_i(w^T\phi(x_i) + b) = 1 \; \forall \; i \in \mathcal{S}$ y $\alpha_i = 0 \; \forall \; i \notin \mathcal{S}$. El bias solo depende de los predictores a través del kernel:

$$b = \frac{1}{N_S}\sum_{i\in\mathcal{S}}\Big(y_i - w^T\phi(x_i)\Big) = \frac{1}{N_S}\sum_{i\in\mathcal{S}}\left(y_i - \sum_{j\in\mathcal{S}}\alpha_j y_j \phi(x_j)^T\phi(x_i)\right)$$

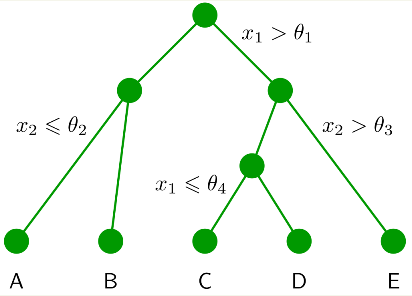
Regla de decisión

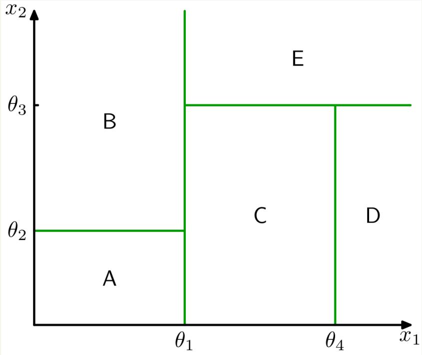
Una vez entrenado, la regla de decisión es simplemente evaluar el signo de $z(x)$. Dicha decisión solo depende de los predictores a través del kernel:

$$z(x) = \sum_{j\in\mathcal{S}}\alpha_j y_j \phi(x_j)^T\phi(x) + b$$

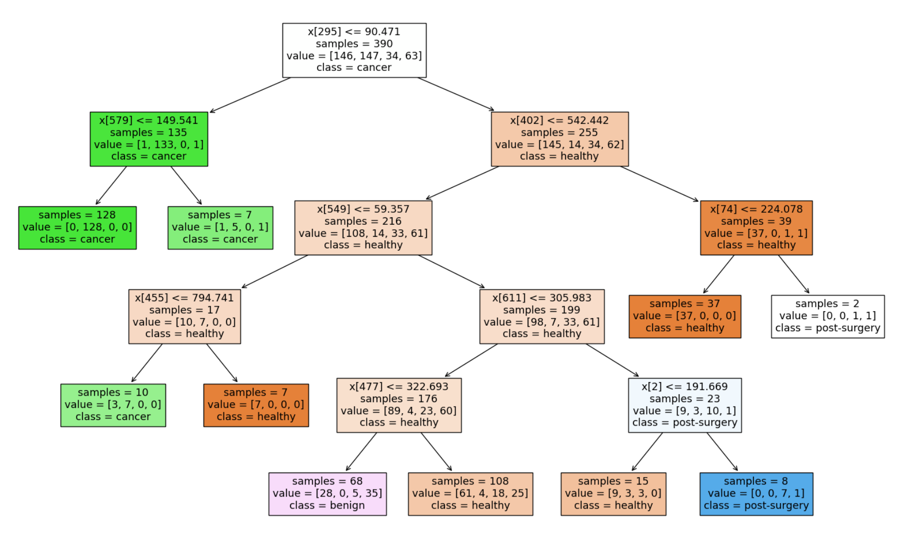
Outline

CART: Classification and Regression Trees





Árboles de decisión



Árboles de decisión

Modelado matemático por nodo

Llamamos:

- Q_m al conjunto de datos en el nodo m.
- $Q_m^L(j_m, t_m) = \{(x, y) \in Q_m : \ x_{j_m} \leq t_m\}$.
- $Q_m^R(j_m, t_m) = \{(x, y) \in Q_m : \ x_{j_m} > t_m\}$.
- $H(Q_m)$ a la función impureza del conjunto Q_m.
- $G_m(j_m, t_m) = \frac{|Q_m^L(j_m,t_m)|}{|Q_m|} H(Q_m^L(j_m, t_m)) + \frac{|Q_m^R(j_m,t_m)|}{|Q_m|} H(Q_m^R(j_m, t_m))$.
- Busco para cada nodo $(j_m^*, t_m^*) = \arg\min_{j_m, t_m} G_m(j_m, t_m)$

Árboles de decisión

Modelado matemático por nodo

Llamamos:

- Q_m al conjunto de datos en el nodo m.
- $Q_m^L(j_m, t_m) = \{(x, y) \in Q_m : \ x_{j_m} \leq t_m\}$.
- $Q_m^R(j_m, t_m) = \{(x, y) \in Q_m : \ x_{j_m} > t_m\}$.
- $H(Q_m)$ a la función impureza del conjunto Q_m.
- $G_m(j_m, t_m) = \frac{|Q_m^L(j_m,t_m)|}{|Q_m|} H(Q_m^L(j_m, t_m)) + \frac{|Q_m^R(j_m,t_m)|}{|Q_m|} H(Q_m^R(j_m, t_m))$.
- Busco para cada nodo $(j_m^*, t_m^*) = \arg\min_{j_m, t_m} G_m(j_m, t_m)$

Funciones impurezas habituales

Sea $p_{m,k}$ la proporción de muestras de la clase k en el nodo m:

- Gini: $H(Q_m) = \sum_k p_{m,k} (1 - p_{m,k})$.
- Entropía: $H(Q_m) = \sum_k -p_{m,k} \log_2 (p_{m,k})$.

Árboles de decisión

Condiciones de Parada

- Todas las observaciones tienen la misma etiqueta.
- Si la rama tiene menos de un número preestablecido de observaciones.
- Poda.

Árboles de decisión

Condiciones de Parada

- Todas las observaciones tienen la misma etiqueta.
- Si la rama tiene menos de un número preestablecido de observaciones.
- Poda.

No se Normaliza

Dado que los árboles trabajan comparando de a un feature por vez, no tiene sentido normalizar. Además al no utilizar gradientes para la optimización, no hay problemas de convergencia.

Árboles de decisión

Condiciones de Parada

- Todas las observaciones tienen la misma etiqueta.
- Si la rama tiene menos de un número preestablecido de observaciones.
- Poda.

No se Normaliza

Dado que los árboles trabajan comparando de a un feature por vez, no tiene sentido normalizar. Además al no utilizar gradientes para la optimización, no hay problemas de convergencia.

Importancia de cada Feature

La *Feature Importance* de un feature se define como la suma de las ganancias, en impureza, de cada nodo (Δ_m) donde se haya utilizado dicho feature para el split con $\Delta_m = H(Q_m) - G_m(j_m^*, t_m^*)$. Habitualmente el resultado se expresa normalizado para que sumen 1.

Árboles de decisión

Problemas de regresión

Modelando la función regresión como constante por regiones, este método puede ser adaptado. Como función impureza suele usarse el error cuadrático medio:

$$H(Q_m) = \sum_{(x,y)\in Q_m} (y - \bar{y}_m)^2$$

donde $\bar{y}_m$ es el promedio de las y en Q_m.

Árboles de decisión

Problemas de regresión

Modelando la función regresión como constante por regiones, este método puede ser adaptado. Como función impureza suele usarse el error cuadrático medio:

$$H(Q_m) = \sum_{(x,y)\in Q_m} (y - \bar{y}_m)^2$$

donde $\bar{y}_m$ es el promedio de las y en Q_m.

Podado: Regularización

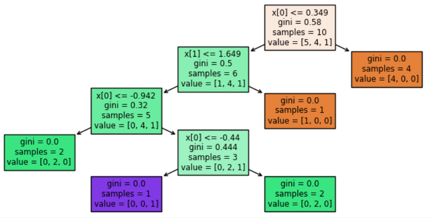
Sea T un árbol determinado (sin condiciones de parado fuertes), $L(T)$ su respectivo conjunto de hojas y α el parámetro de complejidad. Se denomina medida de costo-complejidad a

$$H_\alpha(T) = \sum_{m\in L(T)} \frac{|Q_m|}{n} \cdot H(Q_m) + \alpha \cdot |L(T)|$$

La poda se basa en quedarse con el subárbol de menor costo-complejidad.

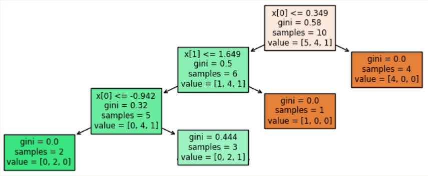
Poda

$\mathbf{T_1}$



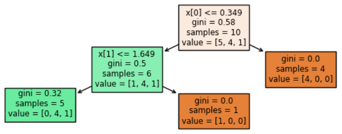
$$H_\alpha(T) = 0 + 5\alpha$$

$\mathbf{T_2}$



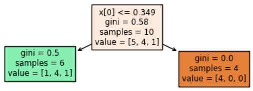
$$H_\alpha(T) = 0.13 + 4\alpha$$

$\mathbf{T_3}$



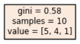
$$H_\alpha(T) = 0.16 + 3\alpha$$

$\mathbf{T_4}$



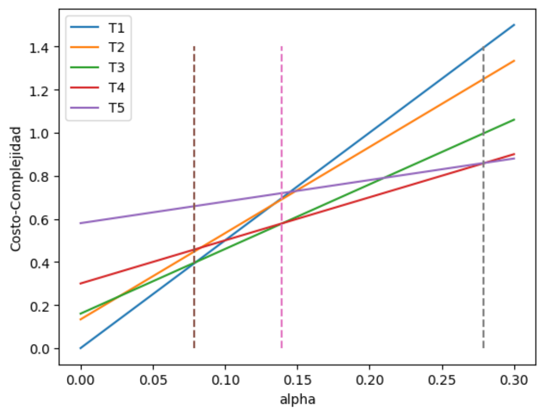
$$H_\alpha(T) = 0.3 + 2\alpha$$

$\mathbf{T_5}$



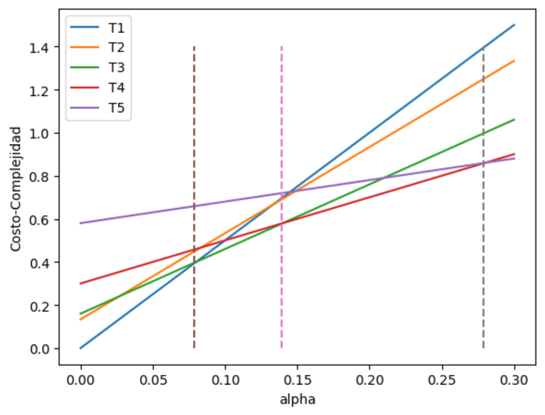
$$H_\alpha(T) = 0.58 + \alpha$$

Poda



T1
T2
T3
T4
T5
Costo-Complejidad
alpha
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

Poda



- La cantidad de candidatos a óptimos es menor a la cantidad de subárboles (el T_2 nunca es el de menor costo-complejidad).
- El subárbol se elige por validación (típicamente sobre el error de clasificación) comparando todos los casos posibles (en este caso 4 candidatos).

Bosques aleatorios

Bagging

El problema de los árboles de decisión es el *overfitting*. Las condiciones de stop y la poda ayudan a combatirlo, pero muchas veces no son suficiente. Es por eso que surge *Bagging*: Entrenar múltiples algoritmos y decidir por mayoría o promedio (en clasificación o regresión respectivamente). Un algoritmo de múltiples árboles se llama bosque.

¿Por que promediar?

- El promedio mantiene la esperanza y reduce la varianza en muestras i.i.d:
$$\mathbb{E}\left[\frac{1}{B}\sum_{b=1}^{B} Z_b\right] = \mu, \qquad \text{var}\left(\frac{1}{B}\sum_{b=1}^{B} Z_b\right) = \frac{\sigma^2}{B}$$
- En clasificación, si se piensan etiquetas en codificación *one-hot*, promediar para luego elegir el máximo equivale a elegir la respuesta mayoritaria.

Bosques aleatorios

Se desea entrenar varios algoritmos (de manera que sean variados). Para asegurar ésto, se toman dos decisiones:

No usar todos los features

En lugar de usar todos los d_x *features*, para asegurar variedad en los árboles, para cada nodo se eligen al azar $\sqrt{d_x}$ *features*.

Bosques aleatorios

Se desea entrenar varios algoritmos (de manera que sean variados). Para asegurar ésto, se toman dos decisiones:

No usar todos los features

En lugar de usar todos los d_x *features*, para asegurar variedad en los árboles, para cada nodo se eligen al azar $\sqrt{d_x}$ *features*.

Bootstrap

Generar B conjuntos de datos diferentes del mismo tamaño que el dataset original n. Para esto, se utiliza una técnica llamada Bootstrap: Se eligen al azar n datos del conjunto *con reposición* y se arma cada conjunto Bootstrap, de manera que la probabilidad que un dato no esté en el conjunto es del $\approx 37\%$:

$$\left(1 - \frac{1}{n}\right)^n \to e^{-1}$$